Assignment 3

Edvin Magnusson

2022-11-07

Question 1

Function for matrix calculation.

```r
my_matrix_prod <- function(A,B){
  if(nrow(A)!=ncol(B)){
    stop("Matrix dimensions mismatch")
  }

  Mat<- matrix(nrow = nrow(A),ncol = ncol(B))

  for(i in 1:nrow(A)){
    for(j in 1:ncol(B)){

      Mat[i,j]<- sum(A[i,]*B[,j])
    }
  }
  return(Mat)

}



X <- matrix(1:6, nrow = 2, ncol = 3)
Y <- matrix(6:1, nrow = 3, ncol = 2)

my_matrix_prod(A=X,B=Y)
```

```
##      [,1] [,2]
## [1,]   41   14
## [2,]   56   20
```

Question 2

Sum of random dice.

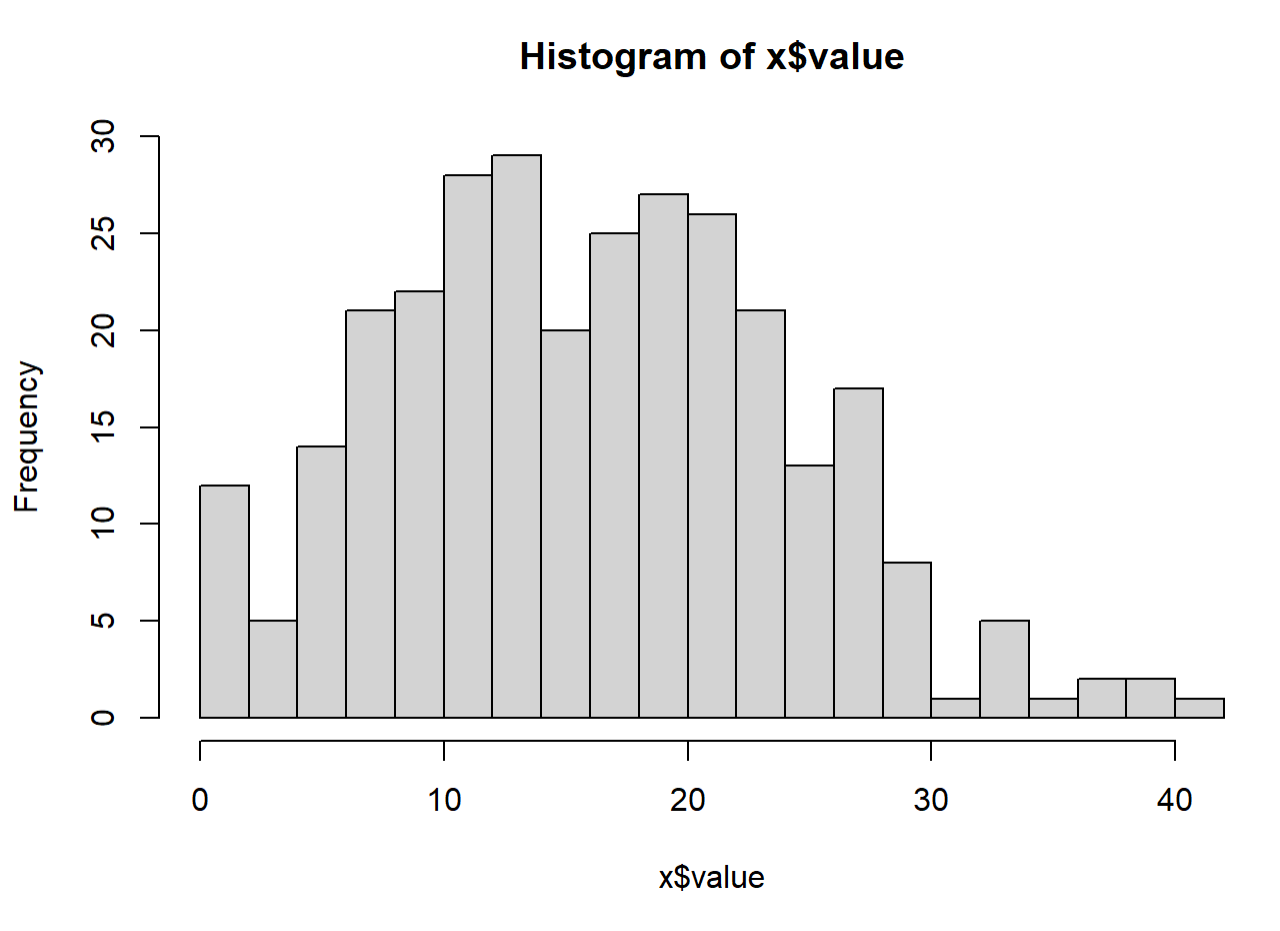
```r
sum_of_random_dice <- function(K,lambda,my_seed=NULL){

  set.seed(my_seed)

  names<-c("value","dice")
  result<- data.frame(matrix(nrow = K,ncol = 2))
  colnames(result)<-names

  current_number <- integer(K)
  sum_value<- integer(K)

  for(k in 1:K){
    current_number[k]<- rpois(1,lambda)
    sum_value[k]<- sum(sample(1:6,size=current_number[k],replace = TRUE))
    result$dice<- current_number
    result$value<- sum_value
  }
  return(result)

}


sum_of_random_dice(K = 5, lambda = 3, my_seed = 42)
```

```
##   value dice
## 1    13    5
## 2     8    4
## 3    18    6
## 4    26    6
## 5    10    4
```

```r
sum_of_random_dice(K = 5, lambda = 8, my_seed = 4711)
```

```
##   value dice
## 1    47   13
## 2    21    5
## 3    30    8
## 4    28    9
## 5    20    8
```

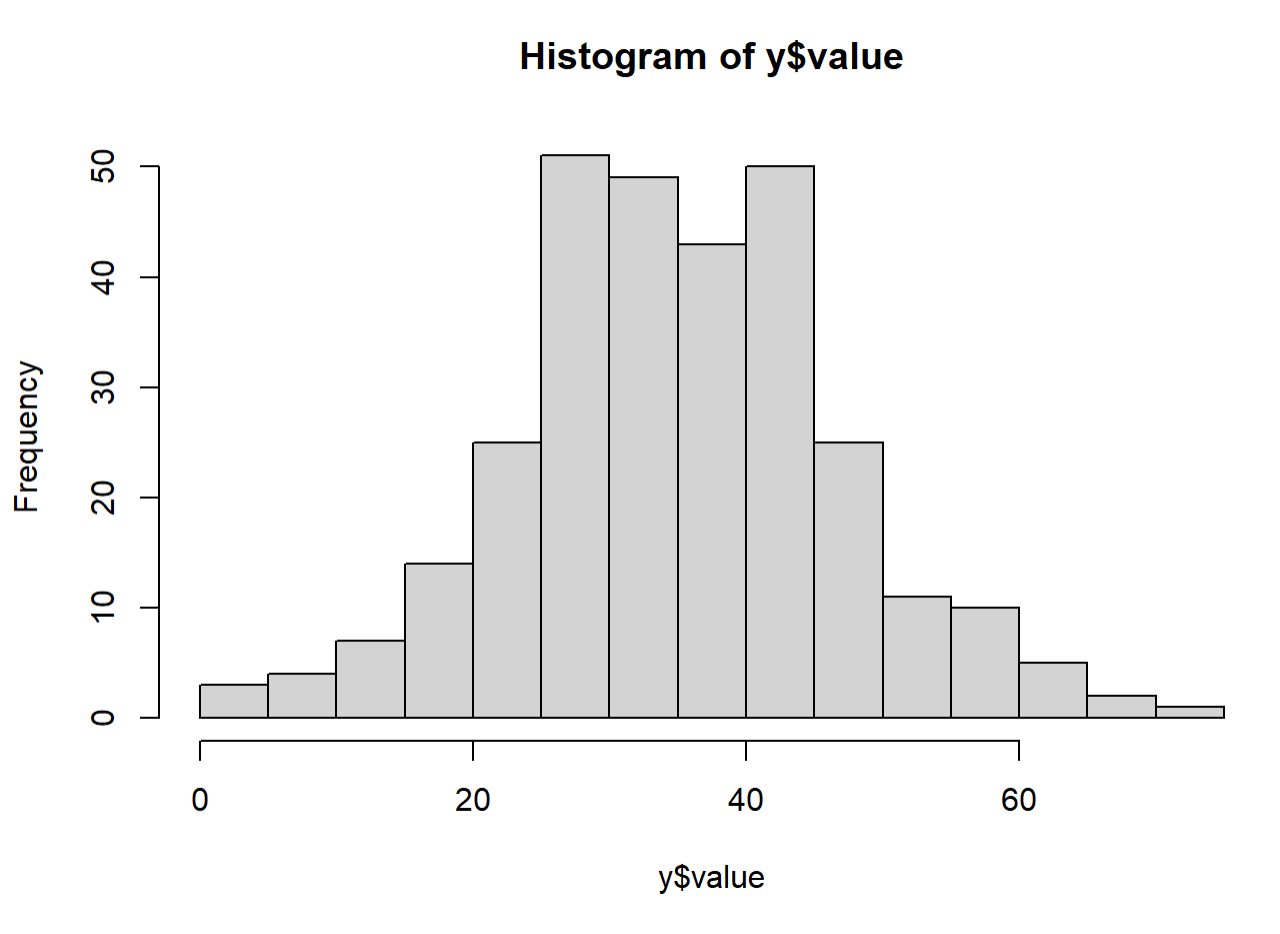
```r
x <- sum_of_random_dice(K = 300, lambda = 5, my_seed = 42)
hist(x$value, 20)
```




```r
mean(x$value)
```

```
## [1] 16.61
```

```r
y <- sum_of_random_dice(K=300, lambda=10, my_seed=4)
hist(y$value, 20)
```



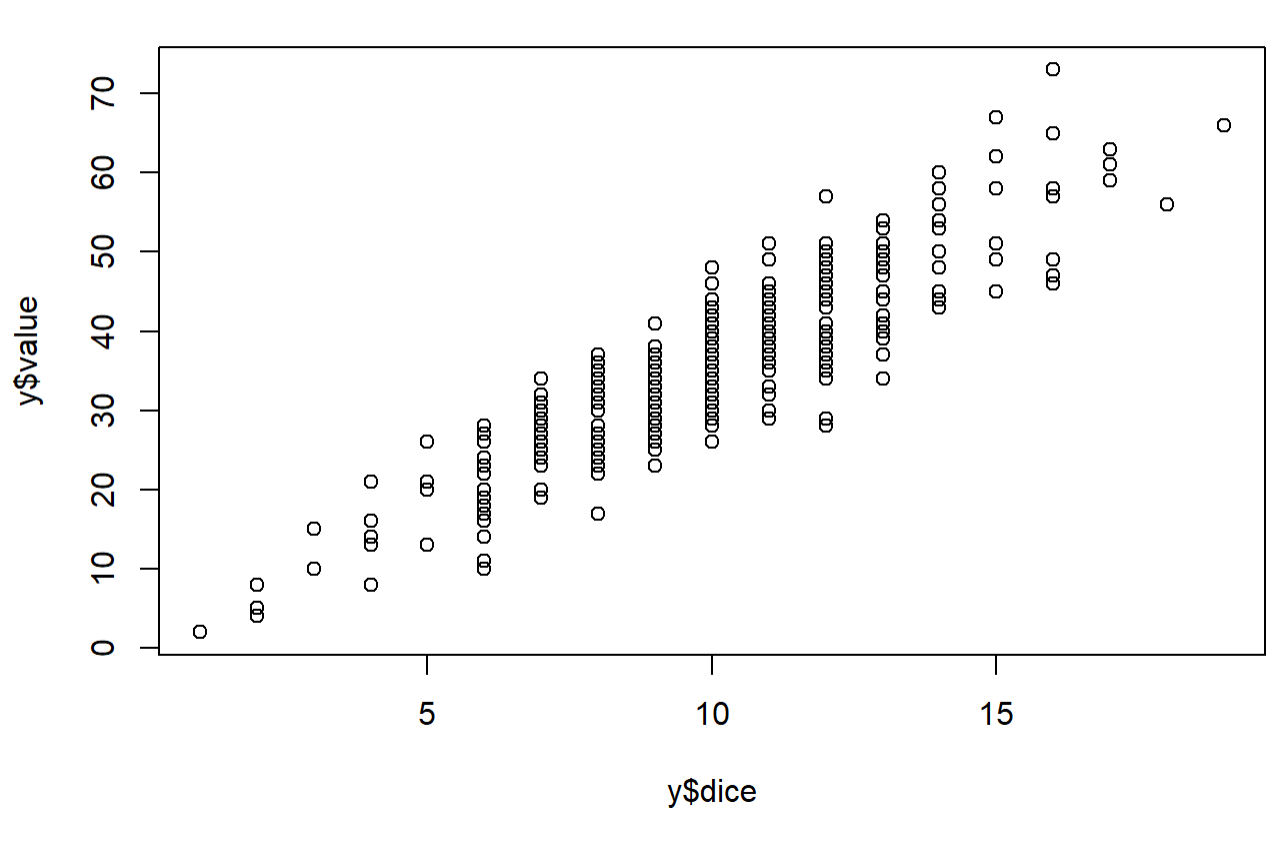

```r
mean(y$value)
```

```
## [1] 35.55333
```

```r
sd(y$value)
```

```
## [1] 12.00169
```

```r
plot(y$dice, y$value)
```



Question 3

Manually creating ols-regression

```r
data("attitude")

my_ols<-function(X,y){

  X<-as.matrix(X)
  X<-cbind(1,X)
  colnames(X)[1]<-"(intercept)"

  XT<- t(X)
  XM<-XT%*%X
  InvX<-solve(XM)
  beta_hat<- InvX%*%XT%*%y
  Y_hat<- X%*%beta_hat
  e_hat<- y-Y_hat
  n<-nrow(X)
  p<-ncol(X)
  sigma2_hat<- (t(e_hat)%*%e_hat)/(n-p)
  my_list<-list(beta_hat=beta_hat,sigma2_hat=sigma2_hat,e_hat=e_hat)
  class(my_list)<-"my_ols"

  return(my_list)

}

data(attitude)
X <- attitude[, 2:4]
y <- attitude[, 1]
inherits(my_ols(X, y), "my_ols")
```

```
## [1] TRUE
```

```r
class(my_ols(X, y))
```

```
## [1] "my_ols"
```

```r
my_ols(X, y)[1:2]
```

```
## $beta_hat
##                   [,1]
## (intercept) 11.2583051
## complaints   0.6824165
## privileges  -0.1032843
## learning     0.2379762
##
## $sigma2_hat
##          [,1]
## [1,] 47.10063
```

```r
head(my_ols(X, y)[["e_hat"]])
```

```
##             [,1]
## [1,]  -9.2440913
## [2,]   0.4838202
## [3,]   2.5755101
## [4,]   0.2123648
## [5,]   6.5906959
## [6,] -11.2012376
```

```r
data(trees)
trees_ols <- my_ols(X = trees[, 1:2], y = trees[, 3])
trees_ols[1:2]
```

```
## $beta_hat
##                    [,1]
## (intercept) -57.9876589
## Girth         4.7081605
## Height        0.3392512
##
## $sigma2_hat
##          [,1]
## [1,] 15.06862
```

```r
summary(trees_ols[[3]])
```

```
##        V1
##  Min.   :-6.4065
##  1st Qu.:-2.6493
##  Median :-0.2876
##  Mean   : 0.0000
##  3rd Qu.: 2.2003
##  Max.   : 8.4847
```

Question 4

Function for confidence intervals.

```r
HUS <- read.csv("C:/Users/edvin/OneDrive/Skrivbord/R Statistik Master/HUS.csv")

# Small corrections (removing outliers)

index <- HUS[, 1] < quantile(HUS[, 1])[4]
HUS <- HUS[index,]




my_grouped_test <- function(data_vector, my_groups, alpha) {
  groups <- as.factor(my_groups)
  result <-  matrix(nrow = length(levels(groups)), ncol = 4)
  colnames(result) <- c("Lower CI-limit", " Mean", "Upper CI-limit", "No of obs.")
  lev <- levels(groups)
  rownames(result) <- lev
  tests <- (by(data_vector,list(groups), t.test, conf.level = 1-alpha))
  num <- as.vector(table(groups))

  lower <-  vector()
  upper <-  vector()
  mean <- vector()
  obs <- vector()

    for (i in 1:length(levels(groups))){
  lower[i] <- tests[[i]][4][1]$conf.int[1]
  upper[i] <- tests[[i]][4][1]$conf.int[2]
  mean[i] <- as.vector(tests[[i]][5][1]$estimate[1])
  obs[i] <- num[i]
  result[i,] <- cbind(lower[i], mean[i], upper[i] , obs[i])
}
return(result)
}




my_grouped_test(HUS[,1], HUS[,5], 0.01)
```

```
##   Lower CI-limit     Mean Upper CI-limit No of obs.
## 0       161467.5 173473.2       185478.8         82
## 1       213182.2 220556.0       227929.7        308
```

```r
my_grouped_test(HUS[,1], HUS[,4], 0.01)
```

```
##   Lower CI-limit     Mean Upper CI-limit No of obs.
## 1       150512.1 159533.1       168554.1         71
## 2       187988.3 196071.4       204154.5        168
## 3       238597.9 247692.7       256787.5        131
## 4       254188.1 280080.6       305973.1         18
## 5      -436567.4 200000.0       836567.4          2
```

```r
my_grouped_test(HUS[,2], HUS[,5], 0.01)
```

```
##   Lower CI-limit     Mean Upper CI-limit No of obs.
## 0       1637.026 1750.805       1864.584         82
## 1       1952.930 2015.026       2077.122        308
```

```r
my_grouped_test(HUS[,8], HUS[,7], 0.01)
```

```
##   Lower CI-limit     Mean Upper CI-limit No of obs.
## 0       1959.597 1961.774       1963.952        372
## 1       1961.980 1967.889       1973.797         18
```

Question 5

Function that helps blood donors when they can give blood again.

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
Sys.setlocale("LC_TIME", "English")
```

```
## [1] "English_United States.1252"
```

```r
give_blood<-function(lasttime,holiday,sex,type_of_travel){

   extratime<-lasttime
   if(holiday!="hemma"){
  if (type_of_travel=="other"){
    extratime<- ymd(int_end(holiday))+weeks(4)+1
  }
  if (type_of_travel=="malaria"){
    extratime<- ymd(int_end(holiday))+months(6)+1
  }}

  if (sex=="m"){
    suggestion<- lasttime+months(3)
  }
  if (sex=="f"){
    suggestion<-lasttime+months(4)
  }

  if (extratime > suggestion){
    proposal<-extratime
  }
  if(extratime < suggestion){
    proposal<-suggestion
    }

  if (wday(proposal)==1){
    proposal<-proposal+days(1)
  }
  if(wday(proposal)==7){
    proposal<-proposal+days(2)
  }
  return(paste("year=",year(proposal),"month=",month(proposal),
                         "day=",day(proposal),"weekday=", weekdays(proposal)))
}

# Setting the date when the donor last gave blood.

day1<-ymd("2014-02-24")

# The date when the donor can give blood again given that the donor is a male  that has not traveled.

give_blood(lasttime=day1, holiday="hemma", sex="m", type_of_travel=NULL)
```

```
## [1] "year= 2014 month= 5 day= 26 weekday= Monday"
```

```r
# The date when the donor can give blood again given that the donor is a female  that has not traveled.

give_blood(lasttime=day1, holiday="hemma", sex="f", type_of_travel=NULL)
```

```
## [1] "year= 2014 month= 6 day= 24 weekday= Tuesday"
```

```r
# If the donor is a male that has been on a holiday trip in a country with malaria and needs some quarantine tim
e.

day2 <- ymd("2014-03-23")
day3 <- ymd("2014-04-24")
holiday1 <- interval(day2, day3)
give_blood(lasttime=day1, holiday=holiday1, sex="m", type_of_travel="malaria")
```

```
## [1] "year= 2014 month= 10 day= 27 weekday= Monday"
```

```r
# If the donor is a female that has been on a holiday trip in a country without malaria and needs some quarantine
time.

day4 <- ymd("2014-04-13")
day5 <- ymd("2014-05-23")
holiday2 <- interval(day4, day5)
give_blood(lasttime=day1, holiday=holiday2, sex="f", type_of_travel="other")
```

```
## [1] "year= 2014 month= 6 day= 24 weekday= Tuesday"
```

Question 6

Checking social security numbers.

```r
# 6.1

  # Check if the last number in the social security number is correct

  pnr_ctrl <-  function(pnr) {
    pnr_split <- as.numeric(strsplit(as.character(pnr), "")[[1]])
    pnr_splits <- pnr_split[3:(length(pnr_split)-1)]
    pnr_odd_double <- (pnr_splits[seq(length(pnr_splits)) %% 2 == 1])*2
    pnr_odd_double <- as.numeric(strsplit(as.character(as.numeric(paste(pnr_odd_double, collapse = ""))), "")[[1
]])
    pnr_even <- pnr_splits[seq(length(pnr_splits)) %% 2 == 0]
    total_sum <- sum(c(pnr_odd_double, pnr_even))
    total_split <-as.numeric(strsplit(as.character(total_sum), "")[[1]])
    control <- 10 - total_split[length(total_split)]
    control <- ifelse(control == 10, 0, control)
    control <- ifelse(control == pnr_split[length(pnr_split)], TRUE, FALSE)
    return(control)
  }

# If TRUE then the last number is correct.

pnr_ctrl(190303030303)
```

```
## [1] FALSE
```

```r
pnr_ctrl(198112189876)
```

```
## [1] TRUE
```

```r
pnr_ctrl(190101010101)
```

```
## [1] FALSE
```

```r
pnr_ctrl("190101010101")
```

```
## [1] FALSE
```

```r
pnr_ctrl("196408233234")
```

```
## [1] TRUE
```

```r
# Check the gender:

  pnr_sex <- function(pnr){
    Male <- (as.numeric(substr(pnr,11,11)) %% 2 == 1) # Use the 11th element,
                                                       # if odd male, if even female
    gender <- factor(ifelse(Male, "Man", "Kvinna"))
    return(gender)
}

pnr_sex("196408233234")
```

```
## [1] Man
## Levels: Man
```

```r
pnr_sex("190202020202")
```

```
## [1] Kvinna
## Levels: Kvinna
```

```r
pnr_sex(190202020202)
```

```
## [1] Kvinna
## Levels: Kvinna
```